Sinais

Segunda aula

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007

Conteúdo

Sinais e **fork**

Propriedades herdadas pelo processo filho

O processo fillho herda a configuração de sinais do processo pai. Nota: os endereços das rotinas de atendimento são válidas no processo filho.

Diferenças entre o processo pai e filho

- o filho começa sem alarmes pendentes
- o conjunto de alarmes pendentes no processo filho é o conjunto vazio

Sinais e **exec**

Sumário

Quando um programa é exec'ed o status de todos os sinais é default ou ignore. Normalmente todos os sinais passam a executar a sua acção por omissão, a não ser que o processo que invoque exec esteja a ignorar o sinal.

Propriedades herdadas pelo processo *execed*

- o tempo que falta para o próximo alarme
- máscara de sinais
- sinais pendentes

Herança de configurações

```
1  void sigterm_handler(int signo)
2  {
3      ...
4  }
5
6  int main(void)
7  {
8      signal(SIGTERM, sigterm_handler);
9
10     for (i = 0; i < NFILHOS; i++) {
11         p[i] = fork();
12         if (p[i] == 0) {
13             /* Processo filho herda configuracao de sinais do
14              * processo pai.  Para repor accao por omissao:
15              *   signal(SIGTERM, SIG_DFL);  */
16             ...
17             exit(0);
18         }
19     }
20     ...
21 }
```

Conteúdo

Sinal **SIGTERM**

Descrição

- Sinal utilizado para provocar a finalização de programas. Ao contrário do SIGKILL, este sinal pode ser bloqueado, interceptado e ignorado.
- Maneira delicada de pedir a um programa que termine.
- Sinal enviado por omissão pelo comando **kill(1)**.

Exemplo de intercepção do sinal **SIGTERM**

```
1  volatile int flag;
2
3  void handler(int signo)
4  {
5      flag = 1;
6  }
7
8  int main(void)
9  {
10     if (signal(SIGTERM, handler) == SIG_ERR) {
11         perror("signal");  exit(SIGTERM);
12     }
13
14     printf("Eu sou o processo %d\n", getpid());
15     while (!flag) {
16         pause();
17     }
18
19     printf("Fim\n");  return 0;
20 }
```

Sinal **SIGCHLD**

Descrição

Sempre que um processo termina ou pára, é enviado ao processo pai um sinal SIGCHLD. Por omissão, o sinal é ignorado.

Processo pai

O processo pai deve interceptar o sinal SIGCHLD se pretender ser notificado de alterações de estado de processos filhos. O tratamento típico deste consiste em invocar uma das chamadas ao sistema da família **wait** para obter o identificador do processo (pid) e a sua informação de estado.

Exemplo de intercepção do sinal **SIGCHLD** (extracto)

```
1  void sigchld_handler( int signo )
2  {
3      ...
4      while ( (res = waitpid(-1, &status, WNOHANG)) > 0) {
5          if (WIFEXITED(status)) {
6              /* o processo terminou normalmente */
7          } else {
8              /* o processo nao terminou normalmente */
9          }
10     }
11 }
12
13 int main(void)
14 {
15     if (signal(SIGCHLD, sigchld_handler) == SIG_ERR) {
16         perror("signal");  exit(SIGCHLD);
17     }
18     ...
19 }
```

Exemplo de intercepção do sinal **SIGCHLD** (extracto)

Preservar valor de **errno**

```
1  void sigchld_handler( int signo )
2  {
3      ...
4      int  olderrno = errno;    /* guardar valor actual de errno */
5
6      while ( (res = waitpid(-1, &status, WNOHANG)) > 0) {
7          if (WIFEXITED(status)) {
8              /* o processo terminou normalmente */
9          } else {
10             /* o processo nao terminou normalmente */
11         }
12     }
13     ...
14     errno = olderrno;         /* repor valor inicial de errno */
15 }
```

Exemplo de intercepção do sinal **SIGCHLD** (extracto)

```
1  #define NUM_FILHOS 4
2
3  void sigchld_handler( int signo )
4  {
5      pid_t res;  int status;
6
7      while ( (res = waitpid(-1, &status, WNOHANG)) > 0) {
8          if (WIFEXITED(status)) {
9              printf("O processo %d terminou normalmente (%d)\n",
10                 res, WEXITSTATUS(status));
11         } else {
12             printf("O processo %d nao terminou normalmente\n", res);
13         }
14     }
15 }
16
17 int main(void)
18 {
19     int i;  signal(SIGCHLD, sigchld_handler);
20
21     for (i = 0; i < NUM_FILHOS; i++) {
22         if (fork() == 0) {  sleep(1); exit(i);  }
23     }
24
25     while (1) {  pause();  }
26
27     return 0;
28 }
```

Exemplo de intercepção do sinal **SIGCHLD** (extracto)

```
1  #define NUM_FILHOS 4
2  volatile int nf;
3
4  void sigchld_handler( int signo )
5  {
6      pid_t res;  int status;
7
8      while ( (res = waitpid(-1, &status, WNOHANG)) > 0) {
9          nf++;
10         if (WIFEXITED(status)) {
11             printf("O processo %d terminou normalmente (%d)\n",
12                    res, WEXITSTATUS(status));
13         } else {
14             printf("O processo %d nao terminou normalmente\n", res);
15         }
16     }
17 }
18
19 int main(void)
20 {
21     int i;  signal(SIGCHLD, sigchld_handler);
22
23     for (i = 0; i < NUM_FILHOS; i++) {
24         if (fork() == 0) {  sleep(1); exit(i);  }
25     }
26
27     while (nf < NUM_FILHOS) {  pause();  }
28
29     printf("Fim\n"); return 0;
30 }
```

Sinais **SIGSTOP SIGCONT**

Sinal **SIGSTOP**

O sinal SIGSTOP pára um processo. Este sinal não pode ser interceptado ou ignorado.

Sinal **SIGCONT**

O sinal SIGSTOP pára um processo. Este sinal não pode ser interceptado ou ignorado.

Conteúdo

Exercício

Exercício

Utilizar vários processos filhos (workers) para verificar a existência de um dado número num vector.
Assumir:

- um vector de mil inteiros
- o número inteiro a procurar
- quatro processos trabalhadores (filhos)

Modificações a realizar:

1. substituir o ciclo for de waits pela recepção do sinal SIGCHLD
2. os processos filhos devem enviar ao processo pai o sinal SIGUSR1 no caso de encontarem o número ou o sinal SIGUSR2 em caso contrário.

Exercício

Downloads concorrentes

Delegar as operações de download em processos filhos (wget ou lftpget). Assumir:

- um array de strings com os links dos ficheiros a descarregar

Funcionalidades:

1. re-lançar downloads no caso do processo filho (wget) ter sido morto ou tiver terminado com um código de saída diferente de zero
2. impor um limite ao número de downloads a decorrer simultaneamente
3. impor um limite temporal para todos os downloads; cancelar todos os downloads pendentes caso o limite seja atingido

Conteúdo

Referências

Bibliografia

- **Advanced Programming in the UNIX Environment, 2nd ed.**
  W. Richard Steven, Stephen A. Rago
  `http://www.apuebook.com/`
  - Capítulo 8 - Process Control
  - Capítulo 10 - Signals
- **Linux Programming by Example: The Fundamentals**
  Arnold Robbins
  `http://authors.phptr.com/robbins/`
  - Capítulo 10 - Signals
- **The Design of the Unix Operating System**
  Maurice J. Bach
  - Capítulo 7 - Secção 7.2 - Signals